CLAIMS

1. An extensible architecture for kernel-mode processing of audio messages, the architecture comprising:

a plurality of modules coupled together, in a module graph implemented in kernel-mode, each of the modules performing an operation in the processing of the audio messages; and

a plurality of additional modules that can be added to the module graph, wherein each of the plurality of additional modules is coupled to one of the plurality of modules.

2. The extensible architecture of claim 1, wherein the audio messages include audio data.

3. The extensible architecture of claim 1, wherein the audio messages include control information.

4. The extensible architecture of claim 1, further comprising a graph builder component, communicatively coupled to the plurality of modules, to add selected ones of the plurality of additional modules to the module graph.

5. The extensible architecture of claim 1, wherein one of the plurality of modules comprises a packer module to communicate the audio messages to one or more of a plurality of different applications executing in user-mode.

6. The extensible architecture of claim 1, wherein one of the plurality of modules comprises an unpacker module to receive the audio messages from one or more of a plurality of different applications executing in user-mode.

7. The extensible architecture of claim 1, wherein each of the plurality of modules in the module graph and each of the plurality of additional modules that is added to the module graph is configured with a pointer to a common reference clock.

8. The extensible architecture of claim 1, wherein one of the plurality of modules is a sequencer module to reorder the audio messages by a timestamp corresponding to the messages, and to delay forwarding the audio messages to another module in the module graph until an appropriate time.

9. The extensible architecture of claim 8, wherein the sequencer module is to forward an audio message to another module at a time that allows the audio message to arrive at a destination hardware device an amount of time prior to a presentation time corresponding to the audio message.

10. The extensible architecture of claim 9, wherein the amount of time is identified by the destination hardware device.

11. The extensible architecture of claim 1, wherein the audio data includes MIDI (Musical Instrument Digital Interface) data.

12. The extensible architecture of claim 1, wherein selected ones of the plurality of additional modules are added to the module graph in response to user-inputs identifying desired audio processing functionality.

13. One or more computer-readable media having stored thereon a sequence of instructions that, when executed by one or more processors of a computer, causes the one or more processors to perform acts including:

identifying a plurality of modules to be used in a module graph to process, in kernel-mode, audio data in a user-identified manner; and

configuring the plurality of modules to process the audio data.

14. One or more computer-readable media as recited in claim 13, wherein the configuring comprises adding additional modules from a module library to the module graph.

15. One or more computer-readable media as recited in claim 13, wherein the configuring comprises re-configuring connected outputs of the plurality of modules.

16. One or more computer-readable media as recited in claim 13, wherein the configuring comprises invoking a ConnectOutput interface on one of the plurality of modules to identify another of the plurality of modules that output from the one module is to be forwarded to during processing of the audio data.

17. One or more computer-readable media as recited in claim 13, wherein the sequence of instructions, when executed, causes the one or more processors to perform acts further including identifying the plurality of modules in response to input received from an application executing in user-mode.

18. One or more computer-readable media as recited in claim 13, wherein the audio data includes MIDI (Musical Instrument Digital Interface) data.

19. One or more computer-readable media as recited in claim 13, wherein the plurality of modules are further configured to process audio control information.

20. One or more computer-readable media as recited in claim 19, wherein audio control information includes one or more of: a volume change, a pan change, and a 3D coordinate change.

21. One or more computer-readable media as recited in claim 13, wherein the configuring comprises passing, to each of the plurality of modules, a pointer to a same reference clock.

22. One or more computer-readable media as recited in claim 13, wherein the configuring comprises passing, to each of the plurality of modules, a pointer to a same allocator module.

23. One or more computer-readable media as recited in claim 13, wherein the one or more processors perform the acts in kernel-mode.

24. A system comprising:

a plurality of modules that can be coupled in various combinations to process, at a privileged level, audio data; and

a graph builder, communicatively coupled to the plurality of modules, to connect together selected ones of the plurality of modules to process the audio data in a particular manner.

25. A system as recited in claim 24, wherein the privileged level comprises a kernel-mode.

26. A system as recited in claim 24, wherein the particular manner comprises a user-identified manner.

27. A system as recited in claim 24, wherein the graph builder is also executed at the privileged level.

28. A computer-readable medium having stored thereon a data structure, the data structure comprising:

a presentation time portion indicating when audio data is to be rendered;

a data portion that can include audio data or a pointer to a chain of additional data structures that include the audio data; and

a flag portion indicating to a kernel-mode transform filter whether the data portion includes the pointer to the chain of additional data structures.

29. A computer-readable medium as recited in claim 28, wherein the data structure further comprises a structure byte count portion that identifies the size of the data structure.

30. A computer-readable medium as recited in claim 28, wherein the data structure further comprises an event byte count portion that identifies a number of data bytes that are referred to in the data portion.

31. A computer-readable medium as recited in claim 28, wherein the data structure further comprises a channel group portion that identifies which of a plurality of channel groups the data identified in the data portion corresponds to.

32. A computer-readable medium as recited in claim 28, wherein the flags portion further includes an event incomplete flag that can be set to indicate that data identified in the data portion extends beyond a buffer pointed to by a pointer maintained in the data portion.

33. A computer-readable medium as recited in claim 28, wherein the data structure further comprises a byte position portion including an identifier of where the data structure is situated among a plurality of data structures received from an application.

34. A computer-readable medium as recited in claim 28, wherein the data structure further comprises a next event portion including an identifier of a next data structure in a chain of data structures.

35. A computer-readable medium as recited in claim 28, wherein:

the data portion can further include a pointer to a data buffer; and

the flag portion indicates whether the data portion includes either the pointer to the chain of additional data structures or one of either the audio data or the pointer to the data buffer.

36. A computer-readable medium as recited in claim 35, further comprising an event byte count portion that identifies, if the data portion does not include the pointer to the chain of additional data structures, whether the data portion includes the audio data or a pointer to the data buffer.

37. A method of processing audio data in a privileged level module, the method comprising:

receiving a data packet including a pointer to a chain of additional data packets that include audio data.

38. A method as recited in claim 37, wherein the processing further comprises carrying out a programmed operation on each of the additional data packets.

39. A method as recited in claim 37, further comprising detecting that the data packet includes the pointer based on a package event flag included in the data packet.

40. A computer-readable memory containing a computer program that is executable by a computer to perform the method recited in claim 37.

41. A method of processing audio data in a kernel-mode transform filter, the method comprising:

passing, to another kernel-mode transform filter, a data packet including a pointer to a chain of additional packets that include audio data.

42. A method as recited in claim 41, further comprising indicating that the data packet includes the pointer by setting a package event flag included in the data packet.

43. A computer-readable memory containing a computer program that is executable by a computer to perform the method recited in claim 41.

44. A system comprising:

a first module implemented in kernel-mode and coupled to receive audio data from hardware;

a second module implemented in kernel mode and coupled to communicate processed audio data to an application executing in user-mode; and

a third module, implemented in kernel-mode, to receive the audio data form the first module, process the audio data, and communicate the processed audio data to the second module.

45. A system as recited in claim 44, wherein the first module is further to process the audio data before forwarding the audio data to the second module.

46. A system as recited in claim 45, wherein the first module is to process the audio data by obtaining a data packet structure into which the audio data can be placed.

47. A system as recited in claim 44, further comprising additional modules, situated between the first and third modules, to further process the audio data.

48. One or more computer-readable media having stored thereon a series of instructions that, when executed by one or more processors of a computer, causes the one or more processors to perform acts including:

maintaining a pool of memory available for allocation to a plurality of transform filters executing at a privileged level;

allocating a portion of the pool of memory to one of the plurality of transform filters to use to store audio data; and

returning the allocated portion to the pool of memory after the plurality of transform filters have finished processing the audio data.

49. One or more computer-readable media as recited in claim 48, wherein the privileged level comprises kernel-mode.

50. One or more computer-readable media as recited in claim 48, wherein the portion comprises sufficient memory to store a data structure including:

a data portion that can include one of: audio data, a pointer to a chain of additional data structures that include the audio data, and a pointer to a data buffer;

a structure byte count portion that identifies the size of the data structure;

a channel group portion that identifies which of a plurality of channel groups the data identified in the data portion corresponds to;

a presentation time portion indicating when audio data is to be rendered;

a flag portion indicating whether the data portion includes either the pointer to the chain of additional data structures or one of either the audio data or the pointer to the data buffer; and

an event byte count portion that identifies, if the data portion does not include the pointer to the chain of additional data structures, whether the data portion includes the audio data or a pointer to the data buffer.

51. One or more computer-readable media as recited in claim 48, wherein the portion comprises a data buffer to store a plurality of audio data messages.

52. One or more computer-readable media as recited in claim 48, wherein the series of instructions, when executed, further cause the one or more processors to perform acts including requesting additional memory, from a memory manager, to add to the pool of memory.

53. One or more computer-readable media as recited in claim 52, wherein the series of instructions, when executed, further cause the one or more processors to perform acts including requesting additional non-paged memory from the memory manager to add to the pool of memory.

54. A method of initializing a module for processing audio data in kernel-mode, the method comprising:

receiving, from a software component, an indication of a next module in a module graph to which the module is to output data; and

receiving, from the software component, a state identifier indicating that the module is to begin processing data.

55. A method as recited in claim 54, further comprising receiving, from the software component, a set of parameters identifying how the module is to process the audio data.

56. A method as recited in claim 54, wherein the software component is executing in kernel-mode.

57. A method as recited in claim 54, further comprising:

receiving, from the software component, a request for current parameters of the module; and

returning, to the software component, the current parameters of the module.

58. A computer-readable memory containing a computer program that is executable by a computer to perform the method recited in claim 54.

59. One or more computer-readable media having stored thereon a transform filter for execution in kernel-mode that, when executed in kernel-mode by one or more processors of a computer, causes the one or more processors to implement:

a ConnectOutput interface to allow identification to the transform filter of a next transform filter in a transform filter graph to which audio data packets should be communicated by the transform filter; and

a PutMessage interface to allow the audio data packets to be communicated to the next transform filter.

60. One or more computer-readable media as recited in claim 59, wherein the transform filter further causes the one or more processors to implement a SetState interface to allow a state of the transform filter to be set, including a run state and a stop state.

61. One or more computer-readable media as recited in claim 59, wherein the transform filter further causes the one or more processors to implement a DisconnectOutput interface to allow a previously identified next transform filter to be changed.

62. One or more computer-readable media as recited in claim 59, wherein the transform filter further causes the one or more processors to implement a SetParameters interface to allow operational parameters of the transform filter to be set.

63. One or more computer-readable media as recited in claim 59, wherein the transform filter further causes the one or more processors to implement a GetParameters filter to allow operational parameters previously sent to the transform filter to be retrieved.

64. One or more computer-readable media as recited in claim 59, wherein the transform filter further causes the one or more processors to implement a GetMessage interface to allow other transform filters in the transform filter graph to obtain data structures for the audio data packets.

65. One or more computer-readable media as recited in claim 59, wherein the transform filter further causes the one or more processors to implement a GetBufferSize interface to allow other transform filters in the transform filter graph to obtain a size of data buffers allocated by the transform filter.

66. One or more computer-readable media as recited in claim 59, wherein the transform filter further causes the one or more processors to implement a GetBuffer interface to allow other transform filters to obtain data buffers for storage of audio data corresponding to an audio data packet.

67. One or more computer-readable media as recited in claim 59, wherein the transform filter further causes the one or more processors to implement a PutBuffer interface to allow other transform filters to return data buffers to a memory pool for re-allocation.

68. One or more computer-readable media as recited in claim 59, wherein the transform filter comprises a sequencer filter to reorder the audio data packets by timestamp and to delay forwarding the audio packets to the next transform filter until an appropriate time.

69. One or more computer-readable media as recited in claim 59, wherein the transform filter comprises an allocator filter to obtain memory from a memory manager and make portions of the obtained memory available to other transform filters.